WE CLAIM :

1. A method of implementing in a portable manner, fixed-width data types where such fixed-width data types are not directly supported by a programming language, said method comprising the steps of :

a) providing as inputs

(i) a set U of required fixed-width data types that have to be implemented in which each fixed-width data type U_k from the set U has a fixed-data type width of WU_k;

(ii) an ordered set B of basic data types that are directly supported by the said programming language, in which each basic data type B_i from the set B has a data type width WB_i and each data type width WB_{i+1} is greater than or equal to data type width WB_i; and

(iii) a set V having all possible data type widths WV_j for every basic data type B_i from the set B of basic data types;

b) creating a generic data type G with two formal parameters consisting of an integer parameter and a data type parameter;

c) for every combination of data type width WV_j from the set V, and basic data type B_i from the set B creating a specialized generic data type G_{ji} having an integer parameter WVj and a data type parameter B_i and providing a possible implementation within the specialized generic data types G_{ji} for each required fixed-width data type U_k from the set U by comparing the data type width WV_j with data type width WU_k for every required fixed width data types U_k from the set U;

d). (i) if data type width WV_j is equal to the data type width WU_k, implementing the required fixed-width data type U_k by creating and mapping data type U_k to data type B_i;

(ii) if data type width WV_j is greater than the data type width WU_k, implementing the required fixed-width data type U_k by using a sub-range of basic data type B_i;

(iii) if data type width WV_j is lesser than the data type width WU_k and if B_i is not the last basic data type form the set B, implementing the required fixed-width data type U_k by mapping U_k to the implementation of U_k provided by the specialized generic data type G having the integer parameter WB_{i+1} and the data type parameter B_{i+1}; and

(iv) if data type width WV_j is lesser than the data type width WU_k and if B_i is the last basic data type from set B, implementing the required fixed-width data type U_k by using an array with the least required number of elements of basic data type B_i or a record with least required number of fields of basic data type B_i; and

e) finally implementing the set U of required fixed-width data types U_k by selecting from the above possible implementations a correct implementation for each required fixed data type U_k of the set U of required fixed-width data types, by creating and mapping the required fixed-width data type U_k to the implementation of U_k provided by the specialized generic data type G having the integer parameter WB_i and the data type parameter B_i wherein i, j, k and n are all positive integers.

2. A method of implementing in a portable manner, fixed-width data types where such fixed-width data

types are not directly supported by a programming language, said method comprising the steps of :

a) providing as inputs

(i) a set U of required fixed-width data types that have to be implemented in which each fixed-width data type U_k from the set U has a fixed data type width of WU_k;

(ii) an ordered set B of basic data types that are directly supported by the said programming language, in which each basic data type B_i from the set B has a data type width WB_i and each data type width WB_{i+1} is greater than or equal to data type width WB_i; and

(iii) a set V having all possible data type widths WV_j for every basic data type B_i from the set B of basic data types;

b) creating a generic data type G with two formal parameters
consisting of an integer parameter and a data type parameter;

c) for every combination of data type width WV_j from the set V, and basic data type B_i from the set B creating a specialized generic data type G_{ji}

having an integer parameter WVj and a data type parameter B_i and providing a possible implementation within the specialized generic data types G_{ji} for each required fixed-width data type U_k from the set U by comparing the data type width WV_j, with data type width WU_k for every required fixed-width data types U_k from the set U;

d). (i) if data type width WV_j is equal to the data type width WU_k, implementing the required fixed-width data type U_k by creating and mapping data type U_k to data type B_i;

(ii) if data type width WV_j is greater than the data type width WU_k, and if B_i is not the first basic data type from the set B, implementing the required fixed-width data type U_k by creating and mapping the required fixed-width data type U_k to the implementation of U_k provided by the specialised generic data type G having the integer parameter WB_{i-1} and the data type parameter B_{i-1},

(iii) if data type width WV_j is greater than the data type width WU_k and if B_i is the first basic data type form the set B, implementing the required fixed width data type U_k by using a sub-range of basic data type B_i; and

(iv) if data type width WV_j is lesser than the data type width WU_k, implementing the required fixed-width data type U_k by using an array, with the least required number of elements of basic data type B_i or a record, with least required number of fields of basic data type B_i; and

finally implementing the set U of required fixed-width data types U_k by selecting from the above possible implementations a correct implementation for each required fixed-width data type U_k from the set U of required fixed-width data types, by creating and mapping the required fixed-width data type U_k to the implementation of U_k provided by the specialized generic data type G having integer parameter WB_n and the .data type parameter B_n, where B_n being the last basic data type from the set B of basic data types; wherein i, j, k and n are all positive integers.